Laporan Resmi AWS

Praktik Integrasi Berkelanjutan dan Pengiriman Berkelanjutan di AWS

Praktik Integrasi Berkelanjutan dan Pengiriman Berkelanjutan di AWS: Laporan Resmi AWS

Table of Contents

Praktik Integrasi Berkelanjutan dan Pengiriman Berkelanjutan di AWS

Tanggal publikasi: 27 Oktober 2021 (Revisi dokumen)

Abstrak

Laporan ini menjelaskan fitur serta manfaat penggunaan integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD) serta penggunaan alat Amazon Web Services (AWS) di lingkungan pengembangan perangkat lunak Anda. Integrasi berkelanjutan dan pengiriman berkelanjutan adalah praktik terbaik dan merupakan bagian penting dari inisiatif DevOps.

Tantangan pengiriman perangkat lunak

Saat ini, korporasi menghadapi tantangan lanskap persaingan yang berubah dengan cepat, perubahan persyaratan keamanan, dan skalabilitas performa. Korporasi harus menjembatani kesenjangan antara stabilitas operasi dan pengembangan fitur yang cepat. Integrasi berkelanjutan dan pengiriman berkelanjutan (CI/CD) adalah praktik yang memungkinkan perubahan perangkat lunak yang cepat sekaligus mempertahankan stabilitas dan keamanan sistem.

Amazon menyadari sejak awal bahwa tuntutan bisnis untuk memberikan fitur kepada pelanggan retail Amazon.com, anak perusahaan Amazon, dan Amazon Web Services (AWS) memerlukan cara-cara pengiriman perangkat lunak yang baru dan inovatif. Pada skala perusahaan seperti Amazon, ribuan tim perangkat lunak independen harus mampu bekerja sama untuk menghadirkan perangkat lunak dengan cepat, aman, andal, dan bebas waktu penghentian.

Dengan mempelajari cara menghadirkan perangkat lunak dengan kecepatan tinggi, Amazon dan organisasi inovatif lainnya membuat DevOps . DevOps adalah kombinasi filosofi budaya, praktik, dan alat yang mampu meningkatkan kemampuan organisasi untuk menghadirkan aplikasi dan layanan dengan kecepatan tinggi. Dengan prinsip DevOps, organisasi dapat mengembangkan dan meningkatkan produk lebih cepat dari organisasi yang menggunakan proses pengembangan perangkat lunak dan pengelolaan infrastruktur tradisional. Dengan kecepatan ini, organisasi dapat melayani pelanggan dengan lebih baik dan bersaing dengan lebih efektif di pasaran.

Beberapa prinsip ini, seperti tim dua pizza dan arsitektur layanan mikro/berorientasi layanan (SOA), berada di luar cakupan laporan resmi ini. Laporan resmi ini membahas kemampuan CI/CD yang telah dibangun dan ditingkatkan oleh Amazon secara berkelanjutan. CI/CD adalah kunci untuk menghadirkan fitur perangkat lunak dengan cepat dan andal.

AWS kini menawarkan kemampuan CI/CD sebagai suatu rangkaian layanan developer: AWS CodeStar, AWS CodeCommit, AWS CodePipeline, AWS CodeBuild, AWS CodeDeploy, dan AWS CodeArtifact. Developer dan profesional operasi IT yang mempraktikkan DevOps dapat menggunakan layanan ini untuk mengirimkan perangkat lunak dengan cepat dan aman. Bersama-sama, layanan ini membantu menyimpan dan menerapkan kontrol versi ke kode sumber aplikasi Anda dengan aman. Anda dapat menggunakan AWS CodeStar untuk mengatur alur kerja rilis perangkat lunak end-to-end dengan cepat menggunakan layanan ini. Untuk lingkungan yang ada, AWS CodePipeline memiliki fleksibilitas untuk mengintegrasikan setiap layanan secara independen dengan alat yang ada. Layanan-layanan ini memiliki ketersediaan tinggi, mudah diintegrasikan, dan dapat diakses melalui Konsol Manajemen AWS, antarmuka pemrograman aplikasi (API) AWS, dan kit alat pengembangan perangkat lunak (SDK) AWS seperti layanan AWS lainnya.

Apa itu integrasi berkelanjutan dan deployment/pengiriman berkelanjutan?

Bagian ini membahas praktik integrasi berkelanjutan dan pengiriman berkelanjutan serta menjelaskan perbedaan antara pengiriman berkelanjutan dan deployment berkelanjutan.

Integrasi berkelanjutan

Integrasi berkelanjutan (CI) adalah praktik pengembangan perangkat lunak yang memungkinkan developer menggabungkan perubahan kode mereka ke dalam repositori pusat secara rutin, lalu menjalankan pembangunan dan pengujian terotomatisasi. Integrasi berkelanjutan seringnya merujuk pada tahap pembangunan atau integrasi dalam proses rilis perangkat lunak serta memerlukan komponen otomatisasi (misalnya layanan CI atau pembangunan) dan komponen budaya kerja (misalnya belajar agar mampu sering melakukan integrasi). Tujuan utama CI adalah menemukan dan mengatasi bug lebih cepat, meningkatkan kualitas perangkat lunak, dan mengurangi waktu yang dibutuhkan untuk melakukan validasi dan meluncurkan pembaruan perangkat lunak terbaru.

Integrasi berkelanjutan berfokus pada tindakan dan pembaruan kode yang lebih kecil untuk diintegrasikan. Developer mengirim pembaruan kode dengan interval yang sama, setidaknya sekali sehari. Developer menarik kode dari repositori kode untuk memastikan kode pada host lokal digabungkan sebelum mendorong ke server pembuatan. Di tahap ini, server pembuatan menjalankan berbagai pengujian, dan menerima atau menolak pengiriman pembaruan kode.

Tantangan dasar implementasi CI mencakup pengiriman pembaruan yang lebih sering ke basis kode umum, mempertahankan repositori kode sumber tunggal, serta mengotomatiskan pembangunan dan pengujian. Tantangan lainnya meliputi pengujian di lingkungan yang serupa dengan produksi, memberikan visibilitas proses ke tim, dan mempermudah developer dalam mendapatkan versi aplikasi apa pun.

Deployment dan pengiriman berkelanjutan

Pengiriman berkelanjutan (CD) adalah praktik pengembangan perangkat lunak yang perubahan kodenya dibangun, diuji, dan dipersiapkan secara otomatis untuk dirilis ke produksi. CD mengembangkan integrasi berkelanjutan dengan men-deploy semua perubahan kode ke lingkungan pengujian, lingkungan produksi, atau keduanya setelah tahap pembangunan selesai. Pengiriman berkelanjutan dapat sepenuhnya terotomatisasi dengan proses alur kerja atau

terotomatisasi sebagian dengan langkah-langkah manual pada bagian-bagian penting. Jika pengiriman berkelanjutan diimplementasikan dengan benar, developer akan selalu memiliki artefak pembangunan yang siap di-deploy yang telah melewati proses pengujian terstandardisasi.

Dengan deployment berkelanjutan, revisi akan otomatis di-deploy ke lingkungan produksi tanpa persetujuan eksplisit dari developer, menjadikan seluruh proses rilis perangkat lunak terotomatisasi. Hal ini memungkinkan adanya putaran umpan balik pelanggan yang berkelanjutan di awal siklus hidup produk.

Pengiriman berkelanjutan bukan deployment berkelanjutan

Salah satu kesalahpahaman tentang pengiriman berkelanjutan adalah bahwa setiap perubahan yang dilakukan dapat segera diterapkan ke produksi setelah lulus pengujian otomatis. Namun, inti dari pengiriman berkelanjutan bukan untuk segera menerapkan setiap perubahan ke produksi, tetapi untuk memastikan bahwa setiap perubahan siap menuju produksi.

Sebelum men-deploy perubahan ke produksi, Anda dapat menerapkan proses keputusan untuk memastikan bahwa deployment produksi sah dan sudah diaudit. Keputusan ini dapat diambil oleh seseorang, kemudian dieksekusi oleh alat.

Dengan pengiriman berkelanjutan, keputusan untuk melanjutkan proses menjadi suatu keputusan bisnis, bukan teknis. Validasi teknis ada di setiap pengiriman pembaruan.

Meluncurkan perubahan pada produksi bukanlah peristiwa yang mengganggu. Deployment tidak memerlukan tim teknis untuk berhenti mengerjakan rangkaian perubahan berikutnya, dan tidak memerlukan rencana proyek, dokumentasi serah terima, atau jadwal pemeliharaan. Deployment menjadi proses berulang yang telah dilakukan dan diuji berkali-kali dalam lingkungan pengujian.

Manfaat pengiriman berkelanjutan

CD memberikan banyak manfaat bagi tim pengembangan perangkat lunak Anda, termasuk mengotomatiskan proses, meningkatkan produktivitas developer, meningkatkan kualitas kode, dan menghadirkan pembaruan kepada pelanggan lebih cepat.

Mengotomatiskan proses rilis perangkat lunak

CD menyediakan metode yang dapat digunakan oleh tim Anda untuk memeriksa kode yang secara otomatis dibangun, diuji, dan disiapkan untuk dirilis ke produksi sehingga pengiriman perangkat lunak menjadi efisien, tangguh, cepat, dan aman.

Meningkatkan produktivitas developer

Praktik CD membantu produktivitas tim dengan membebaskan developer dari tugas manual, melepaskan dependensi kompleks, dan mengembalikan fokus pada penyediaan fitur baru di perangkat lunak. Developer dapat fokus pada logika pengodean yang mampu memberikan fitur yang Anda butuhkan, bukan pada integrasi kode dengan bagian bisnis lainnya dan menghabiskan waktu untuk menemukan cara men-deploy kode ke platform.

Meningkatkan kualitas kode

CD dapat membantu Anda menemukan dan mengatasi bug di awal proses pengiriman sebelum berkembang menjadi masalah yang lebih besar. Tim Anda dapat melakukan jenis uji kode tambahan dengan mudah karena seluruh proses telah terotomatisasi. Dengan lebih sering memperbanyak pengujian, tim dapat melakukan pengulangan lebih cepat dengan umpan balik langsung pada dampak perubahan. Hal ini memungkinkan tim untuk mendorong kode kualitas dengan jaminan stabilitas dan keamanan tinggi. Developer akan mengetahui apakah kode baru berfungsi dan apakah ada perubahan yang melanggar atau bug melalui umpan balik langsung tersebut. Kesalahan yang teridentifikasi sejak awal dalam proses pengembangan adalah kesalahan yang paling mudah diperbaiki.

Mengirim pembaruan lebih cepat

CD membantu tim Anda mengirim pembaruan kepada pelanggan dengan cepat dan sering. Saat CI/CD diimplementasikan, kecepatan seluruh tim, termasuk rilis fitur dan perbaikan bug akan meningkat.

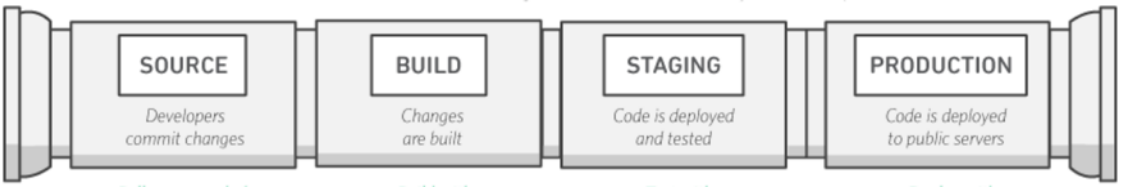
Korporasi dapat lebih cepat merespons perubahan pasar, tantangan keamanan, kebutuhan pelanggan, dan penekanan biaya. Misalnya, jika diperlukan fitur keamanan baru, tim Anda dapat mengimplementasikan CI/CD dengan pengujian otomatis untuk memperkenalkan perbaikan dengan cepat dan andal ke sistem produksi dengan yakin. Hal yang sebelumnya perlu berminggu-minggu atau bulan, sekarang dapat dikerjakan dalam hitungan hari, bahkan jam.

Mengimplementasikan integrasi berkelanjutan dan pengiriman berkelanjutan

Bagian ini membahas cara-cara yang dapat Anda gunakan untuk mulai menerapkan model CI/CD di organisasi Anda. Laporan resmi ini tidak membahas tentang cara suatu organisasi dengan model transformasi cloud dan DevOps yang sudah matang membangun serta menggunakan alur CI/CD. Untuk membantu pengalaman DevOps Anda, AWS memiliki beberapa Partner DevOps bersertifikat yang menyediakan sumber daya dan alat. Untuk informasi lebih lanjut tentang mempersiapkan pindahan ke AWS Cloud, baca Membangun Model Pengoperasian Cloud.

Jalur menuju integrasi berkelanjutan/pengiriman berkelanjutan

CI/CD dapat digambarkan sebagai alur (lihat gambar berikut), yang mana kode baru dikirimkan di satu ujung, diuji melalui serangkaian tahapan (sumber, pembuatan, penahapan, dan produksi), kemudian dipublikasikan sebagai kode yang siap untuk diproduksi. Jika organisasi Anda baru mengenal CI/CD, Anda dapat menerapkan pendekatan berulang pada alur ini. Artinya, Anda harus memulai dari hal kecil, dan mengulang setiap tahapan sehingga Anda dapat memahami dan mengembangkan kode Anda melalui cara yang dapat membantu pertumbuhan organisasi Anda.



Alur CI/CD

Setiap tahapan pada alur CI/CD dirancang sebagai unit logika dalam proses pengiriman. Selain itu, setiap tahapan berfungsi sebagai gerbang yang menyeleksi kode aspek tertentu. Saat kode diproses dalam sebuah alur, dapat diasumsikan bahwa kualitas kode tersebut akan lebih baik di tahap berikutnya karena semakin banyak aspek yang terus diverifikasi. Masalah-masalah yang tidak ditemukan di tahap awal menghentikan pemrosesan kode dalam alur. Hasil pengujian segera dikirim ke tim, dan semua pembangunan serta peluncuran dihentikan jika perangkat lunak gagal melewati tahapan ini.

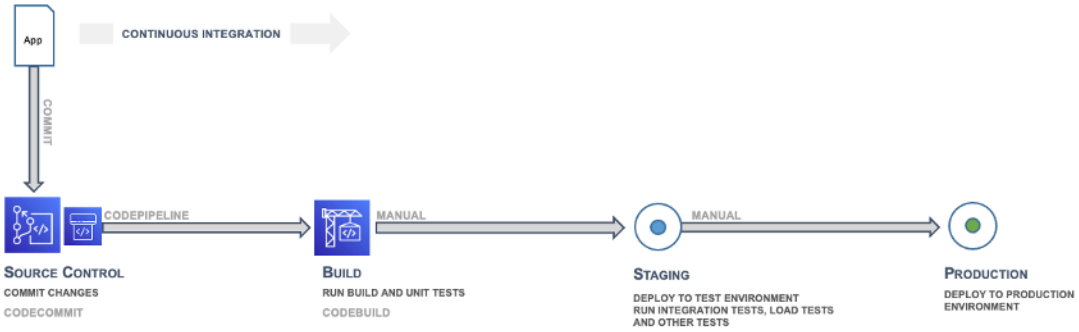
Tahapan-tahapan ini hanya saran. Tahapan ini dapat disesuaikan dengan kebutuhan bisnis Anda. Beberapa tahap dapat diulang untuk beberapa jenis pengujian, keamanan, dan performa. Tergantung pada kompleksitas proyek dan struktur tim Anda, beberapa tahap dapat diulang beberapa kali pada

tingkat yang berbeda. Misalnya, produk akhir dari satu tim bisa menjadi bagian dari proyek tim berikutnya. Artinya, produk akhir tim pertama kemudian dijadikan sebagai artefak dalam proyek tim berikutnya.

Adanya alur CI/CD akan berdampak besar pada pematangan kemampuan organisasi Anda. Organisasi harus mulai dengan langkah-langkah kecil dan tidak membangun alur yang sepenuhnya matang, dengan banyak lingkungan, banyak fase pengujian, dan otomatisasi pada semua tahap di awal. Perlu diingat bahwa bahkan organisasi dengan lingkungan CI/CD yang sangat matang masih harus terus meningkatkan alurnya.

Membangun organisasi dengan kemampuan CI/CD adalah sebuah perjalanan, dan ada banyak tujuan dalam prosesnya. Bagian selanjutnya membahas jalur yang mungkin diambil oleh organisasi Anda, dimulai dengan integrasi berkelanjutan pada tingkatan pengiriman berkelanjutan.

Integrasi berkelanjutan



Integrasi berkelanjutan—menyediakan sumber dan membangun

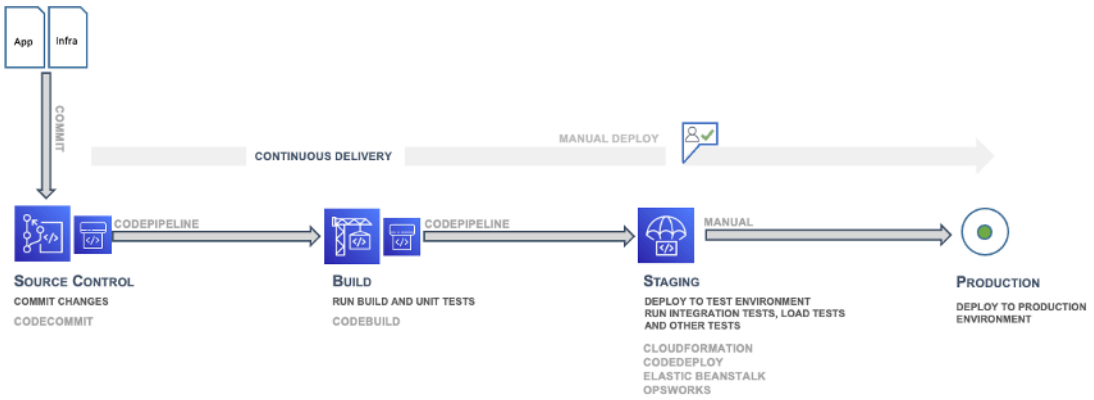
Tahap pertama dalam perjalanan CI/CD adalah mengembangkan kematangan dalam integrasi berkelanjutan. Anda harus memastikan bahwa semua developer mengirim pembaruan kode ke repositori pusat (seperti yang di-host di CodeCommit atau GitHub) secara teratur dan menggabungkan semua perubahan ke cabang rilis untuk aplikasi. Developer dilarang mengisolasi kode. Jika diperlukan cabang fitur untuk jangka waktu tertentu, cabang fitur ini harus terus diperbarui dengan menggabungkan dari upstream sesering mungkin. Agar tim dapat mengembangkan disiplin dan didorong oleh proses, sebaiknya lakukan pengiriman pembaruan kode dengan unit pekerjaan lengkap sesering mungkin. Developer yang menggabungkan kode lebih awal dan sering, kemungkinan akan lebih sedikit menghadapi masalah integrasi ke depannya.

Anda juga harus mendorong developer untuk membuat unit pengujian sedini mungkin untuk aplikasi mereka dan menjalankan pengujian ini sebelum mendorong kode ke repositori pusat. Kesalahan

yang ditemukan di awal proses pengembangan perangkat lunak adalah yang termurah dan paling mudah untuk diperbaiki.

Saat kode didorong ke cabang dalam repositori kode sumber, mesin alur kerja yang memantau cabang ini akan mengirim perintah ke alat pembangun untuk membangun kode dan menjalankan pengujian unit di lingkungan terkendali. Proses pembangunan harus disesuaikan ukurannya dengan tepat untuk menangani semua aktivitas, termasuk dorongan dan pengujian yang mungkin terjadi selama tahap pengiriman pembaruan, untuk mendapatkan umpan balik cepat. Pemeriksaan kualitas lainnya, seperti cakupan uji unit, pemeriksaan gaya, dan analisis statis, juga dapat terjadi di tahap ini. Terakhir, alat pembangun menciptakan satu atau beberapa build biner dan artefak lainnya, seperti gambar, stylesheet, dan dokumen untuk aplikasi.

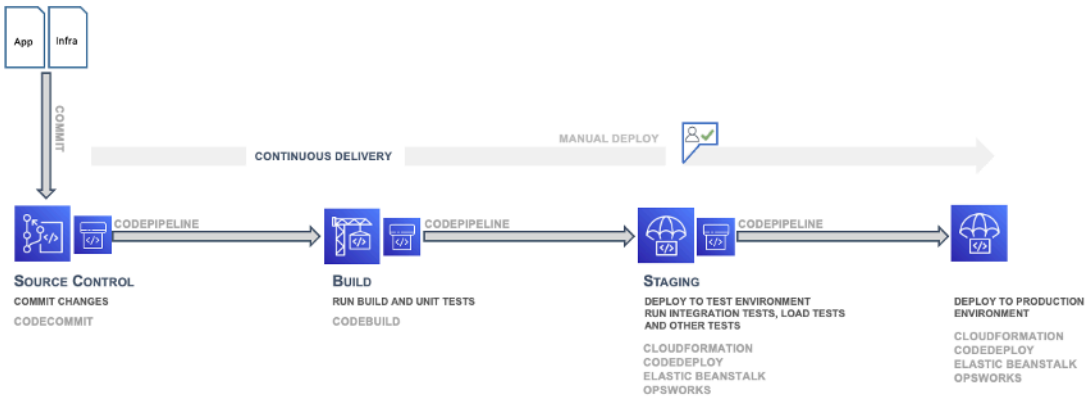
Pengiriman berkelanjutan: membuat lingkungan penahapan



Pengiriman berkelanjutan—penahapan

Pengiriman berkelanjutan (CD) adalah fase berikutnya dan memerlukan penerapan kode aplikasi di lingkungan penahapan, yang merupakan replika tumpukan produksi, dan menjalankan lebih banyak uji fungsional. Lingkungan penahapan bisa menjadi lingkungan statis yang sudah dibuat sebelumnya untuk pengujian, atau Anda dapat menyediakan dan mengonfigurasikan lingkungan dinamis dengan infrastruktur dan kode konfigurasi yang berkomitmen untuk menguji dan men-deploy kode aplikasi.

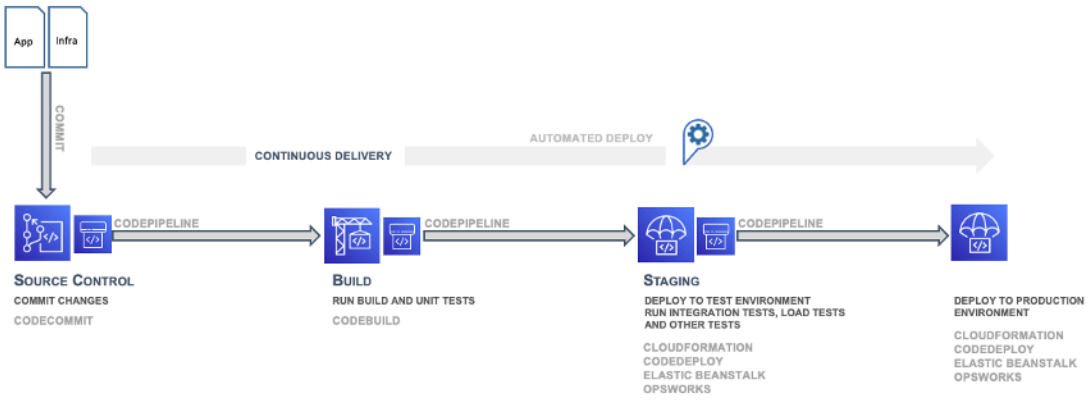
Pengiriman berkelanjutan: membuat lingkungan produksi



Pengiriman berkelanjutan—produksi

Dalam urutan alur pengiriman/deployment, setelah lingkungan penahapan adalah lingkungan produksi, yang juga dibangun menggunakan infrastruktur sebagai kode (IaC).

Deployment berkelanjutan



Deployment berkelanjutan

Tahap akhir dalam alur deployment CI/CD adalah deployment berkelanjutan, yang dapat mencakup otomatisasi penuh pada seluruh proses rilis perangkat lunak, termasuk deployment ke lingkungan produksi. Di lingkungan CI/CD yang sudah sepenuhnya matang, alur menuju lingkungan produksi sudah terotomatisasi penuh, memungkinkan keberhasilan deployment kode.

Lebih dari sekadar kematangan

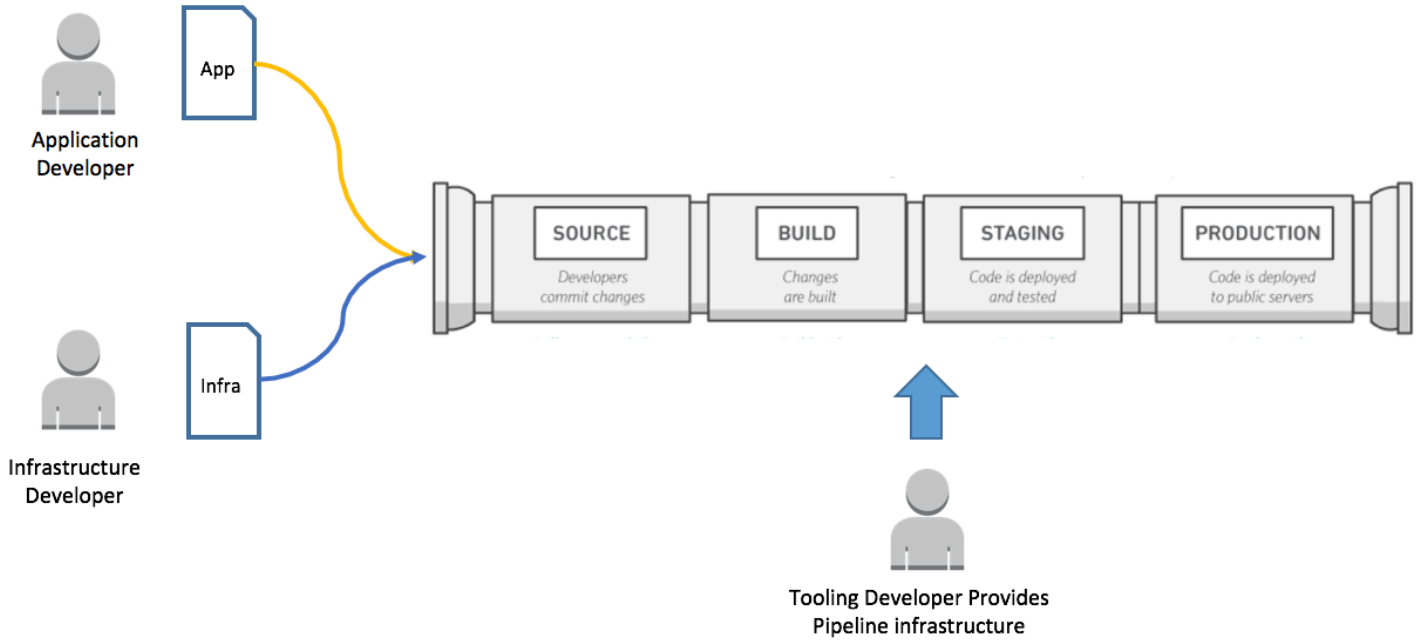
Meskipun sudah matang, organisasi akan terus mengembangkan model CI/CD untuk mencapai peningkatan berikut:

- Lingkungan penahapan tambahan untuk pengujian performa, kepatuhan, keamanan, dan antarmuka pengguna (UI) tertentu
- Uji unit infrastruktur dan kode konfigurasi bersama dengan kode aplikasi
- Integrasi dengan sistem dan proses lain seperti tinjauan kode, pelacakan masalah, dan notifikasi peristiwa
- Integrasi dengan migrasi skema basis data (jika ada)
- Langkah tambahan untuk audit dan persetujuan bisnis

Bahkan organisasi yang paling matang dengan jaringan alur CI/CD multilingkungan yang kompleks pun terus berupaya mencapai peningkatan. DevOps adalah perjalanan, bukan tujuan. Umpan balik terkait alur terus dikumpulkan dan peningkatan kecepatan, skala, keamanan, serta keandalan dicapai sebagai kolaborasi antara berbagai bagian tim pengembangan.

Tim

AWS merekomendasikan untuk membentuk tiga tim developer dalam mengimplementasikan lingkungan CI/CD: tim aplikasi, tim infrastruktur, dan tim alat (lihat gambar berikut). Organisasi ini mewakili serangkaian praktik terbaik yang telah dikembangkan dan diterapkan di perusahaan rintisan yang berkembang pesat, organisasi korporasi besar, dan di Amazon sendiri. Tim harus mengikuti konsep dua pizza, yaitu beranggotakan sekitar 10-12 orang. Hal ini mengikuti aturan komunikasi bahwa percakapan yang efisien akan mencapai batas seiring meningkatnya ukuran grup dan bertambahnya jalur komunikasi.



Tim aplikasi, infrastruktur, dan alat

Tim aplikasi

Tim aplikasi membuat aplikasi. Developer aplikasi memiliki backlog, riwayat, dan pengujian unit, serta mengembangkan fitur berdasarkan target aplikasi tertentu. Tujuan tim ini secara organisasi adalah untuk meminimalkan waktu yang diperlukan oleh para developer untuk selain tugas-tugas aplikasi inti.

Selain memiliki keterampilan pemrograman fungsional dalam bahasa aplikasi, tim aplikasi harus memiliki keterampilan platform dan pemahaman tentang konfigurasi sistem. Keterampilan ini akan membantu mereka untuk fokus hanya pada pengembangan fitur dan perlindungan aplikasi.

Tim infrastruktur

Tim infrastruktur menulis kode yang mampu membuat sekaligus mengonfigurasi infrastruktur yang diperlukan untuk menjalankan aplikasi. Tim ini dapat menggunakan alat AWS, seperti AWS CloudFormation, atau alat generik, seperti Chef, Puppet, atau Ansible. Tim infrastruktur bertanggung jawab menentukan sumber daya apa yang dibutuhkan, dan bekerja sama dengan tim aplikasi. Tim infrastruktur mungkin hanya terdiri dari satu atau dua orang untuk aplikasi kecil.

Tim harus memiliki keterampilan dalam metode penyediaan infrastruktur, seperti AWS CloudFormation atau HashiCorp Terraform. Tim juga harus mengembangkan keterampilan otomatisasi konfigurasi dengan berbagai alat seperti Chef, Ansible, Puppet, atau Salt.

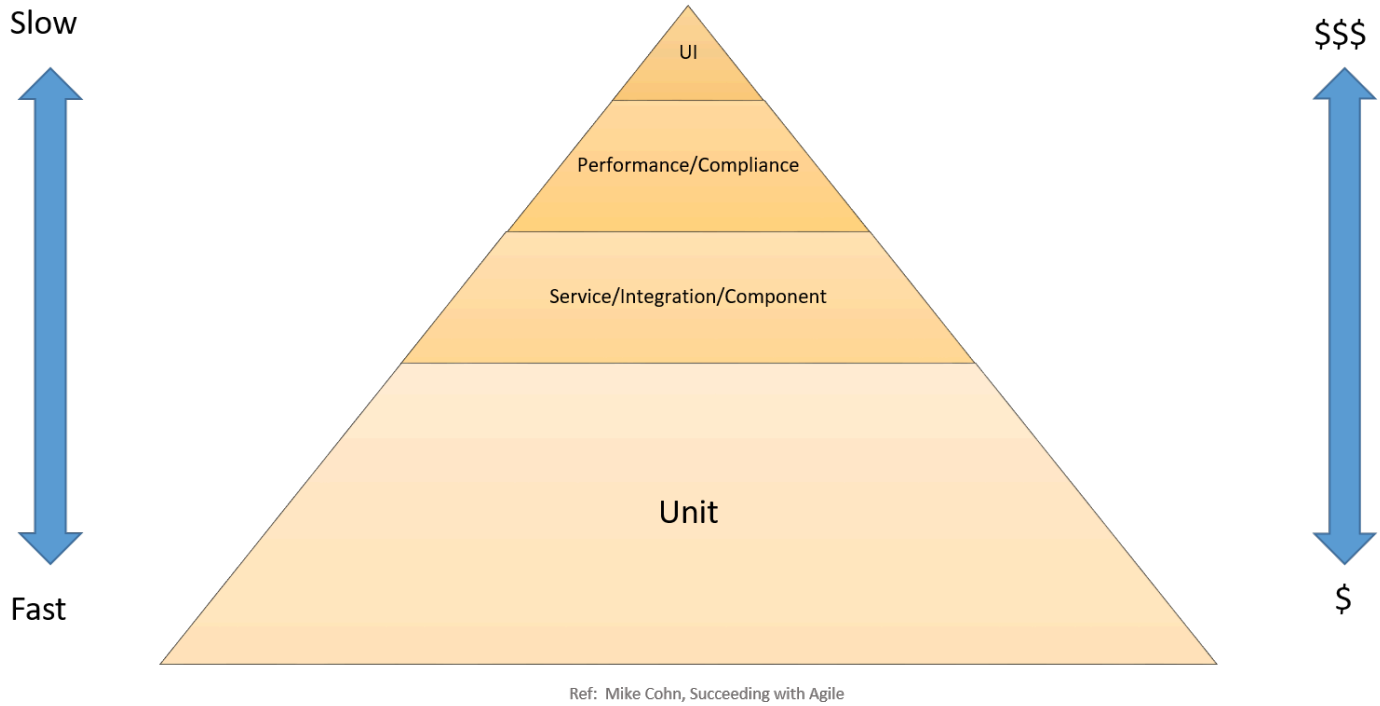
Tim alat

Tim alat membangun dan mengelola alur CI/CD. Mereka bertanggung jawab atas infrastruktur dan alat penyusun alur. Tim ini bukan bagian dari tim dua pizza; tetapi mereka membuat alat yang digunakan oleh tim aplikasi dan infrastruktur di organisasi. Organisasi harus terus mematangkan tim alatnya, sehingga tim alat bisa selangkah lebih maju dari tim aplikasi dan infrastruktur yang juga semakin matang.

Tim alat harus terampil dalam membangun dan mengintegrasikan semua bagian alur CI/CD. Bagian ini meliputi pembangunan repositori kontrol sumber, mesin alur kerja, lingkungan build, pengujian kerangka kerja, dan repositori artefak. Tim ini dapat mengimplementasikan perangkat lunak seperti AWS CodeStar, AWS CodePipeline, AWS CodeCommit, AWS CodeDeploy, AWS CodeBuild, dan AWS CodeArtifact, serta Jenkins, GitHub, Artifactory, TeamCity, dan alat serupa lainnya. Beberapa organisasi menyebutnya tim DevOps, tetapi AWS menekankan bahwa DevOps adalah gabungan antara orang, proses, dan alat dalam pengiriman perangkat lunak.

Tahap pengujian dalam integrasi berkelanjutan dan pengiriman berkelanjutan

Ketiga tim CI/CD harus menggabungkan pengujian ke dalam siklus hidup pengembangan perangkat lunak di berbagai tahap alur CI/CD. Secara keseluruhan, pengujian harus dimulai sedini mungkin. Piramida pengujian berikut merupakan konsep yang disampaikan oleh Mike Cohn dalam Succeeding with Agile. Piramida ini menunjukkan berbagai pengujian perangkat lunak berdasarkan biaya dan kecepatannya.



Piramida pengujian CI/CD

Pengujian unit berada di bagian bawah piramida. Pengujian tersebut merupakan yang paling cepat dijalankan dan yang paling murah. Oleh karena itu, pengujian unit harus menjadi bagian terbesar dari strategi pengujian Anda. Baiknya adalah sekitar 70 persen. Pengujian unit harus memiliki cakupan kode yang hampir lengkap karena bug yang teridentifikasi fase ini dapat diperbaiki dengan cepat dan murah.

Pengujian layanan, komponen, dan integrasi dalam piramida berada di atas pengujian unit. Pengujian ini memerlukan lingkungan yang terperinci, sehingga keperluan infrastrukturnya lebih mahal dan lebih lambat untuk dijalankan. Pengujian performa dan kepatuhan berada pada tingkat di atasnya. Pengujian ini memerlukan lingkungan berkualitas produksi dan lebih mahal lagi. Pengujian UI dan penerimaan pengguna berada di bagian puncak piramida dan juga memerlukan lingkungan berkualitas produksi.

Semua pengujian ini adalah bagian dari strategi lengkap untuk menjamin kualitas perangkat lunak. Namun, demi kecepatan pengembangan, fokusnya adalah pada jumlah pengujian dan cakupan di separuh bagian bahwa piramida.

Bagian berikut membahas tahapan CI/CD.

Menyiapkan sumber

Di awal proyek, penting untuk menyiapkan sumber untuk tempat menyimpan kode mentah serta perubahan konfigurasi dan skema. Di tahap sumber, pilih repositori kode sumber seperti yang di-host di GitHub atau AWS CodeCommit.

Menyiapkan dan menjalankan pembangunan

Otomatisasi pembangunan sangat penting untuk proses CI. Saat menyiapkan otomatisasi pembangunan, tugas pertama adalah memilih alat pembangunan yang tepat. Ada banyak alat pembangunan, di antaranya:

- Ant, Maven, dan Gradle untuk Java
- Make untuk C/C++
- Grunt untuk JavaScript
- Rake untuk Ruby

Alat pembangunan terbaik akan bergantung pada bahasa pemrograman proyek dan set keterampilan tim Anda. Setelah memilih alat pembangunan, semua dependensi harus ditentukan dengan jelas dalam skrip pembangunan, bersama dengan langkah pembangunan. Cara ini juga merupakan praktik terbaik untuk membuat versi artefak pembangunan akhir, sehingga lebih mudah di-deploy dan mengetahui masalah yang ada.

Membangun

Di tahap pembangunan, alat pembangunan akan menjadikan setiap perubahan ke repositori kode sumber sebagai input, membangun perangkat lunak, dan menjalankan jenis pengujian berikut:

Pengujian Unit – Menguji bagian kode tertentu untuk memastikan kode berfungsi sebagaimana mestinya. Pengujian unit dilakukan oleh developer perangkat lunak selama tahap pengembangan. Di tahap ini, analisis kode statis, analisis aliran data, cakupan kode, dan proses verifikasi perangkat lunak lainnya dapat diterapkan.

Analisis Kode Statis – Pengujian ini dilakukan tanpa benar-benar mengeksekusi aplikasi setelah pembangunan dan pengujian unit. Analisis ini dapat membantu menemukan kesalahan pengodean dan celah keamanan, serta mampu memastikan kesesuaian dengan pedoman pengodean.

Penahapan

Di fase penahapan, lingkungan penuh dibuat dan mirip dengan lingkungan produksi sebenarnya. Pengujian yang dilakukan:

Pengujian integrasi – Memverifikasi antarmuka antarkomponen terhadap desain perangkat lunak. Pengujian integrasi merupakan proses berulang serta memfasilitasi pembangunan antarmuka dan integritas sistem yang tangguh.

Pengujian komponen – Menguji pesan yang muncul di antara berbagai komponen dan hasilnya. Tujuan utama dari pengujian ini adalah idempotensi dalam pengujian komponen. Pengujian bisa mencakup data dalam volume yang sangat besar, atau situasi edge dan input abnormal.

Pengujian sistem – Menguji sistem end-to-end dan memverifikasi apakah perangkat lunak memenuhi persyaratan bisnis. Pengujian ini bisa mencakup pengujian antarmuka pengguna (UI), API, logika backend, dan status akhir.

Pengujian performa – Menentukan kemampuan respons dan stabilitas sistem saat dijalankan pada beban kerja tertentu. Pengujian performa juga digunakan untuk menyelidiki, mengukur, memvalidasi, atau memverifikasi atribut kualitas sistem lainnya, seperti skalabilitas, keandalan, dan penggunaan sumber daya. Jenis pengujian performa bisa mencakup uji beban, uji tekanan, dan uji lonjakan. Pengujian performa digunakan untuk pembandingan terhadap kriteria yang telah ditetapkan.

Pengujian kepatuhan – Memeriksa apakah perubahan kode sesuai dengan persyaratan spesifikasi dan/atau peraturan nonfungsional. Pengujian ini menentukan apakah Anda mengimplementasikan dan memenuhi standar yang ditetapkan.

Pengujian penerimaan pengguna – Memvalidasi alur bisnis end-to-end. Pengujian ini dilakukan oleh pengguna akhir dalam lingkungan penahapan dan mengonfirmasi apakah sistem sudah memenuhi spesifikasi persyaratan. Biasanya, di tahap ini pelanggan menerapkan metode pengujian alfa dan beta.

Produksi

Setelah melewati pengujian sebelumnya, fase penahapan diulang di lingkungan produksi. Di fase ini, Pengujian canary akhir dapat diselesaikan dengan men-deploy kode baru pada sebuah subset server kecil atau bahkan satu server, atau satu Wilayah AWS sebelum men-deploy kode ke seluruh lingkungan produksi. Informasi terkait cara men-deploy ke produksi dengan aman tercantum di bagian Metode deployment.

Bagian berikutnya membahas pembangunan alur untuk menggabungkan tahapan dan pengujian ini.

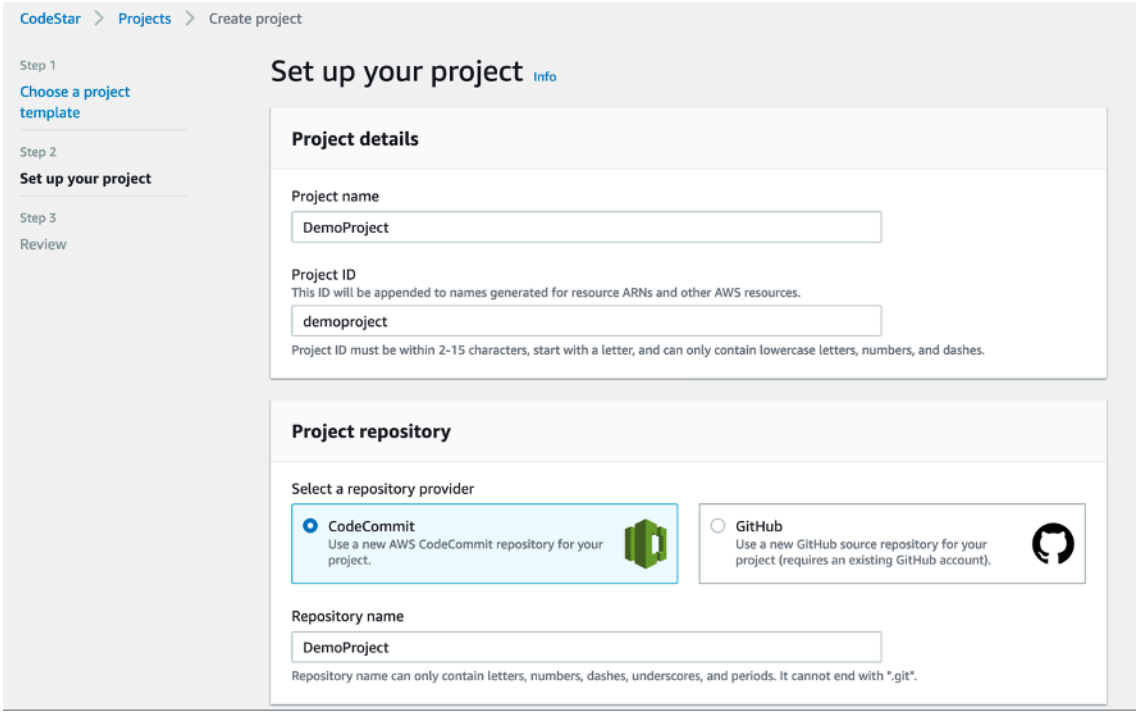
Membangun alur

Bagian ini membahas pembangunan alur. Mulai dengan membuat alur hanya dengan komponen yang diperlukan untuk CI, kemudian nantinya bertransisi ke alur pengiriman berkelanjutan dengan lebih banyak komponen dan tahapan. Bagian ini juga membahas bagaimana Anda dapat menggunakan persetujuan manual dan fungsi AWS Lambda untuk proyek besar, merencanakan banyak tim, cabang, dan Wilayah AWS.

Memulai alur layak minimum untuk integrasi berkelanjutan

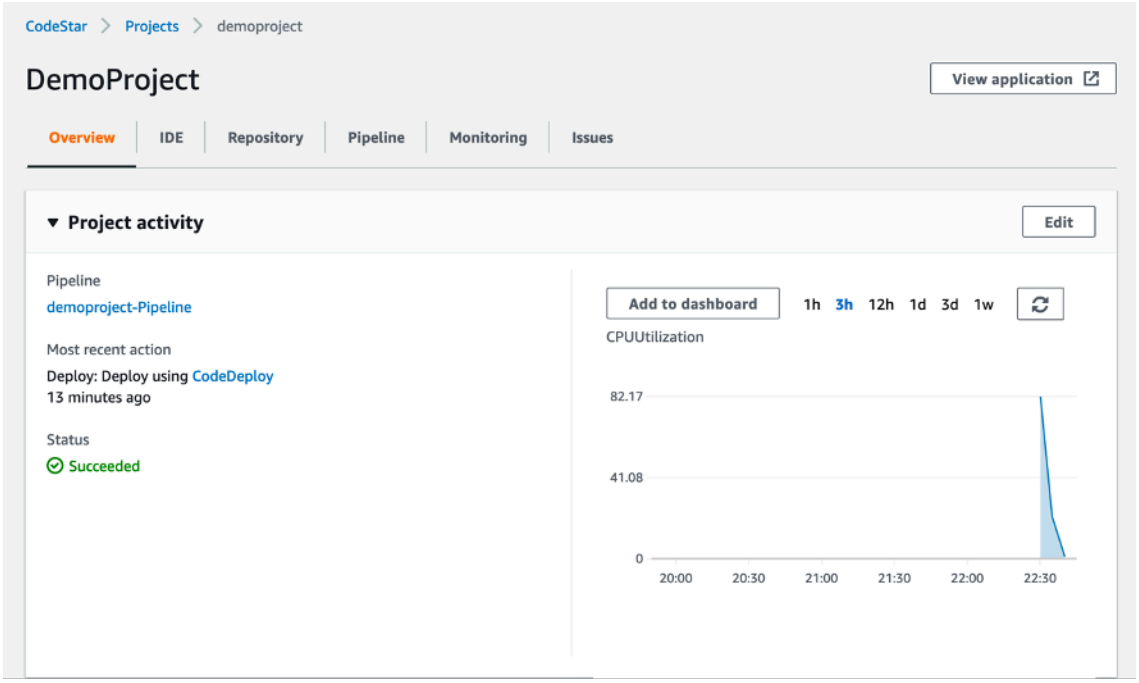
Perjalanan organisasi Anda menuju pengiriman berkelanjutan dimulai dengan alur layak minimum (MVP). Seperti yang dibahas dalam Mengimplementasikan integrasi berkelanjutan dan pengiriman berkelanjutan, tim dapat memulai dengan proses yang sangat sederhana, seperti mengimplementasikan alur yang dapat melakukan pemeriksaan gaya kode atau pengujian unit tunggal tanpa deployment.

Komponen kuncinya adalah alat orkestrasi pengiriman berkelanjutan. Untuk membantu Anda dalam membangun alur ini, Amazon mengembangkan AWS CodeStar.



Halaman penyiapan AWS CodeStar

AWS CodeStar menggunakan AWS CodePipeline, AWS CodeBuild, AWS CodeCommit, dan AWS CodeDeploy dengan proses, alat, templat, dan dasbor penyiapan terintegrasi. AWS CodeStar menyediakan segala yang Anda butuhkan untuk mengembangkan, membangun, dan men-deploy aplikasi di AWS. Hal ini memungkinkan Anda untuk mulai merilis kode lebih cepat. Pelanggan yang sudah terbiasa dengan Konsol Manajemen AWS dan menginginkan tingkat kontrol yang lebih tinggi dapat mengonfigurasikan pilihan alat developer secara manual serta menyediakan layanan AWS individual sesuai kebutuhan.

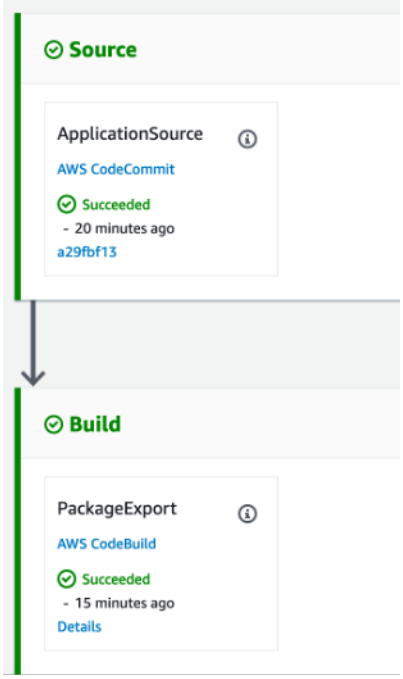


Dasbor AWS CodeStar

AWS CodePipeline adalah layanan CI/CD yang dapat digunakan melalui AWS CodeStar atau Konsol Manajemen AWS untuk menghasilkan pembaruan infrastruktur serta aplikasi yang cepat dan andal. AWS CodePipeline membangun, menguji, dan men-deploy kode jika ada perubahan kode, berdasarkan model proses rilis yang Anda tentukan. Hal ini memungkinkan Anda untuk mengirim fitur serta pembaruan dengan cepat dan andal. Anda dapat membangun solusi end-to-end dengan mudah menggunakan plugin kami yang sudah dibuat sebelumnya untuk layanan pihak ketiga yang populer seperti GitHub atau mengintegrasikan plugin kustom Anda sendiri ke tahap proses rilis Anda. Dengan AWS CodePipeline, Anda hanya perlu membayar sesuai penggunaan. Tidak ada biaya awal atau komitmen jangka panjang.

Langkah AWS CodeStar dan AWS CodePipeline memetakan langsung ke tahap CI/CD penyediaan sumber, pembangunan, penahapan, dan produksi. Tidak perlu langsung memulai dengan pengiriman

berkelanjutan. Anda dapat memulai dengan alur dua langkah sederhana yang memeriksa repositori sumber dan melakukan tindakan pembangunan:

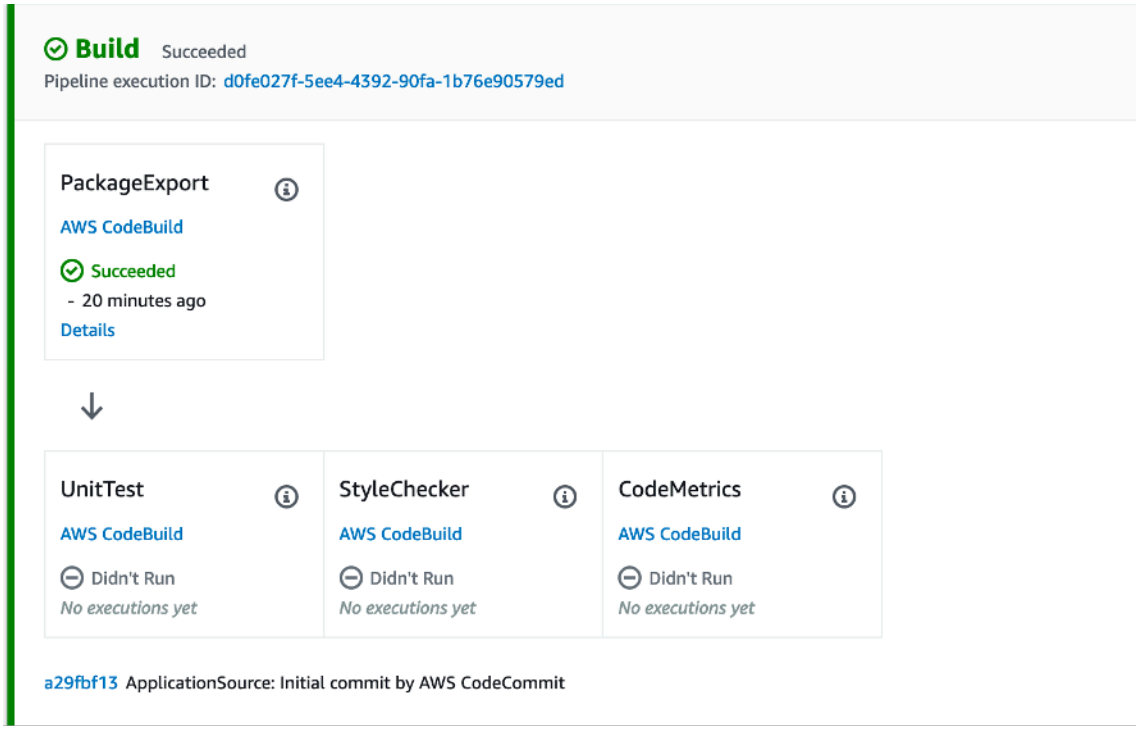


AWS CodePipeline — tahap penyediaan sumber dan pembangunan

Untuk AWS CodePipeline, tahap penyediaan sumber dapat menerima input dari GitHub, AWS CodeCommit, dan Amazon Simple Storage Service (Amazon S3). Otomatisasi proses pembuatan adalah langkah pertama yang penting untuk mengimplementasikan pengiriman berkelanjutan dan deployment berkelanjutan. Tidak adanya keterlibatan manusia dalam memproduksi artefak pembangunan menghilangkan beban tim Anda, meminimalkan kesalahan yang disebabkan oleh pengemasan manual, dan Anda dapat mulai lebih sering mengemas artefak habis pakai.

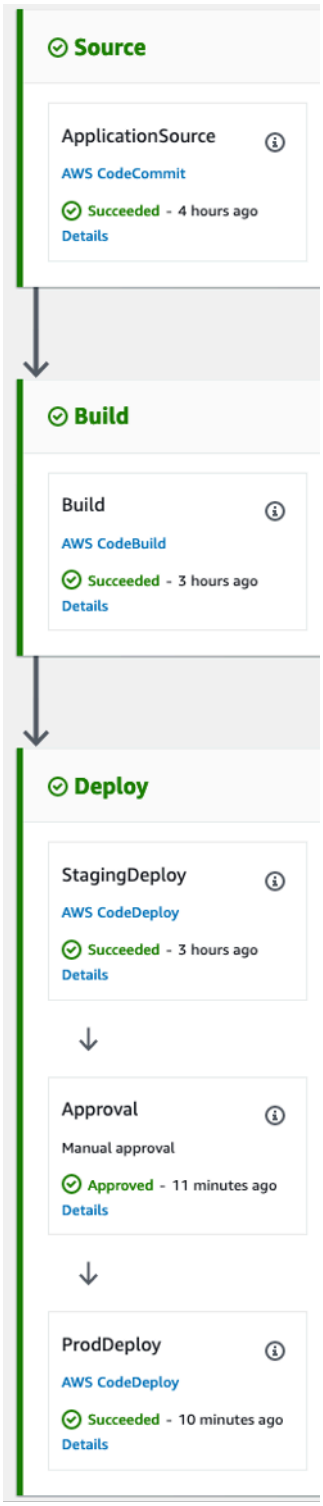
AWS CodePipeline berfungsi baik dengan AWS CodeBuild, layanan pembangunan terkelola penuh, untuk mempermudah penyiapan langkah pembangunan alur serta mampu mengemas kode dan menjalankan pengujian unit. Dengan AWS CodeBuild, Anda tidak perlu menyediakan, mengelola, atau menskalakan server pembuatan Anda sendiri. AWS CodeBuild menskalakan terus-menerus dan memproses beberapa build secara bersamaan sehingga build Anda tidak perlu menunggu dalam antrean. AWS CodePipeline juga terintegrasi dengan server pembuatan seperti Jenkins, Solano CI, dan TeamCity.

Misalnya, dalam tahap pembangunan berikut, tiga tindakan (pengujian unit, pemeriksaan gaya kode, dan pengumpulan metrik kode) berjalan secara paralel. Dengan AWS CodeBuild, langkah-langkah ini dapat ditambahkan sebagai proyek baru tanpa upaya lebih dalam membangun atau menginstal server pembuatan untuk menangani beban.



AWS CodePipeline — fungsionalitas build

Tahap penyediaan sumber dan pembangunan yang ditampilkan pada gambar AWS CodePipeline — tahap sumber dan pembangunan , bersama dengan proses dan otomatisasi pendukung, mendukung perubahan tim Anda menuju Integrasi Berkelanjutan. Di tingkat kematangan ini, developer harus rutin memperhatikan hasil pengujian dan pembangunan. Mereka juga perlu meningkatkan dan memelihara kondisi dasar pengujian unit. Hal ini dapat memperkuat kepercayaan diri seluruh tim dalam alur CI/CD dan penggunaan lainnya.



Tahapan AWS CodePipeline

Alur pengiriman berkelanjutan

Setelah implementasi alur integrasi berkelanjutan dan penetapan proses pendukung, tim Anda dapat mulai bertransisi dalam alur pengiriman berkelanjutan. Dalam transisi ini, tim harus mengotomatiskan deployment dan pembangunan aplikasi.

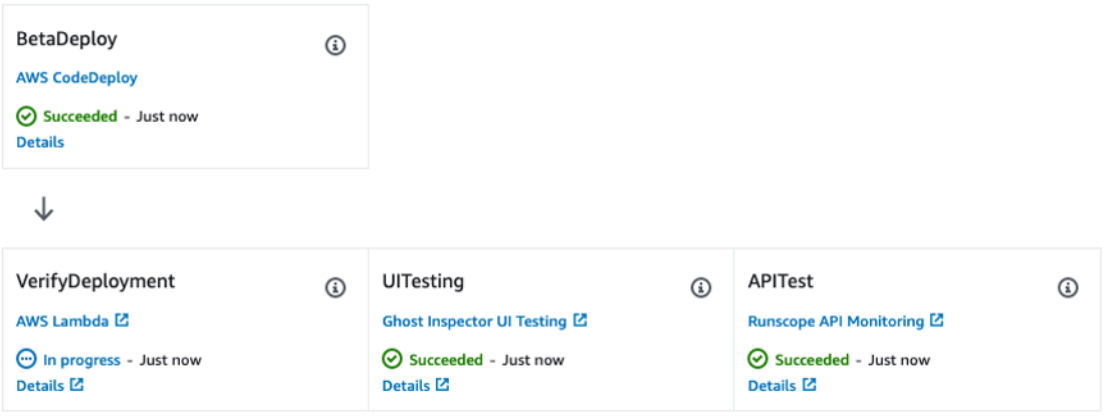
Alur pengiriman berkelanjutan ditandai dengan adanya tahapan dan langkah produksi, yang mana langkah produksi dilakukan setelah adanya persetujuan manual.

Sama seperti pembangunan alur integrasi berkelanjutan, tim Anda bisa mulai membangun alur pengiriman berkelanjutan secara bertahap dengan menuliskan skrip deployment-nya.

Beberapa langkah deployment dapat diabstraksikan dengan layanan AWS yang ada, tergantung pada kebutuhan aplikasi. Misalnya, AWS CodePipeline berintegrasi langsung dengan AWS CodeDeploy, layanan yang mengotomatiskan deployment kode ke instans Amazon EC2 dan instans yang dijalankan secara on-premise, AWS OpsWorks, layanan pengelolaan konfigurasi yang membantu pengoperasian aplikasi menggunakan Chef, dan ke AWS Elastic Beanstalk, layanan untuk men-deploy dan menskalakan layanan dan aplikasi web.

AWS memiliki dokumentasi lengkap tentang cara mengimplementasikan dan mengintegrasikan AWS CodeDeploy dengan alur dan infrastruktur Anda.

Setelah tim Anda berhasil mengotomatiskan deployment aplikasi, tahapan deployment dapat diperluas dengan berbagai pengujian. Misalnya, Anda dapat menambahkan integrasi unik lainnya dengan layanan seperti Ghost Inspector, Runscope, dan lainnya seperti yang ditunjukkan pada gambar berikut.



AWS CodePipeline—pengujian kode dalam tahapan deployment

Menambahkan tindakan Lambda

AWS CodeStar dan AWS CodePipeline mendukung integrasi dengan AWS Lambda. Integrasi ini memungkinkan implementasi serangkaian tugas, seperti membuat sumber daya kustom di lingkungan Anda, mengintegrasikan dengan sistem pihak ketiga (seperti Slack), dan melakukan pemeriksaan pada lingkungan yang baru Anda deploy.

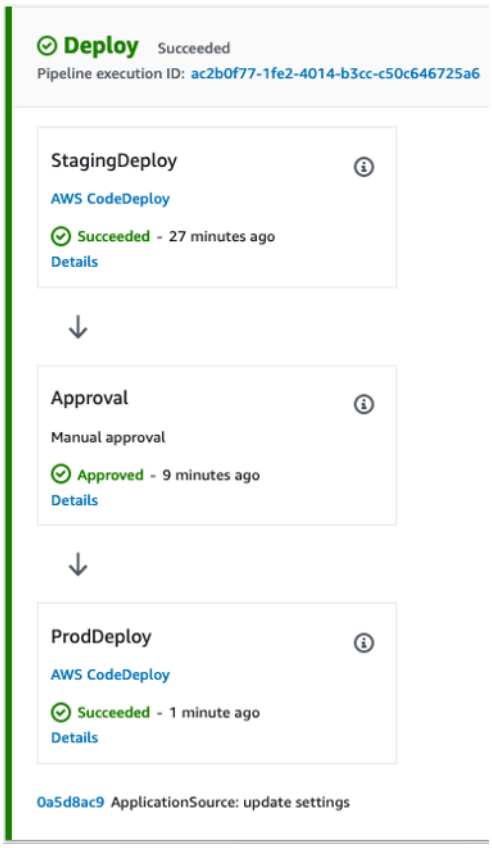
Fungsi Lambda dapat digunakan dalam alur CI/CD untuk melakukan tugas-tugas berikut:

- Membuat perubahan ke lingkungan Anda dengan menerapkan atau memperbarui templat AWS CloudFormation.
- Membuat sumber daya sesuai permintaan dalam satu tahap alur menggunakan AWS CloudFormation dan menghapusnya di tahap lain.
- Men-deploy versi aplikasi dengan waktu henti nol AWS Elastic Beanstalk dengan fungsi Lambda yang menukar nilai catatan Nama Kanonik (CNAME).
- Men-deploy ke instans Docker Amazon Elastic Container Service (ECS).
- Mencadangkan sumber daya sebelum membangun atau men-deploy dengan membuat snapshot AMI.
- Menambahkan integrasi dengan produk pihak ketiga ke alur Anda, seperti mem-posting pesan ke klien Internet Relay Chat (IRC).

Persetujuan manual

Tambahkan tindakan persetujuan ke dalam sebuah tahap pada alur jika Anda ingin menghentikan pemrosesan alur sehingga seseorang dengan izin AWS Identity and Access Management (IAM) yang diperlukan dapat menyetujui atau menolak tindakan tersebut.

Jika tindakan disetujui, pemrosesan alur akan dilanjutkan. Jika tindakan ditolak—atau jika tidak ada orang yang menyetujui atau menolak tindakan ini dalam waktu tujuh hari sejak alur mencapai tindakan ini dan berhenti—tindakan dianggap gagal, dan pemrosesan alur tidak dilanjutkan.



AWS CodeDeploy—persetujuan manual

Men-deploy perubahan kode infrastruktur dalam alur CI/CD

AWS CodePipeline memungkinkan Anda untuk memilih AWS CloudFormation sebagai tindakan deployment dalam tahap alur mana pun. Anda kemudian dapat memilih tindakan spesifik yang ingin Anda AWS CloudFormation lakukan, seperti membuat atau menghapus tumpukan serta membuat atau mengeksekusi set perubahan. Tumpukan merupakan suatu konsep AWS CloudFormation dan mewakili sekelompok sumber daya AWS terkait. Meskipun ada banyak cara penyediaan Infrastruktur sebagai code (IaC), AWS CloudFormation adalah alat komprehensif yang direkomendasikan oleh AWS sebagai solusi lengkap dan dapat diskalakan yang dapat menjelaskan rangkaian sumber daya AWS terlengkap sebagai kode. AWS merekomendasikan untuk menggunakan AWS CloudFormation dalam proyek AWS CodePipeline untuk melacak pengujian dan perubahan infrastruktur.

CI/CD untuk aplikasi nirserver

Anda juga dapat menggunakan AWS CodeStar, AWS CodePipeline, AWS CodeBuild, dan AWS CloudFormation untuk membangun alur CI/CD untuk aplikasi nirserver. Aplikasi nirserver mengintegrasikan layanan terkelola seperti Amazon Cognito, Amazon S3, dan Amazon DynamoDB

dengan layanan yang didorong peristiwa, serta AWS Lambda untuk men-deploy aplikasi tanpa perlu pengelolaan server. Jika Anda adalah developer aplikasi nirserver, Anda dapat menggunakan kombinasi antara AWS CodePipeline, AWS CodeBuild, dan AWS CloudFormation untuk mengotomatiskan pembangunan, pengujian, serta deployment aplikasi nirserver yang ditunjukkan dalam templat yang dibuat dengan AWS Serverless Application Model. Untuk informasi lebih lanjut, lihat dokumentasi AWS Lambda untuk Mengotomatiskan Deployment Aplikasi Berbasis Lambda.

Anda juga dapat membuat jaringan alur CI/CD yang aman sesuai praktik terbaik organisasi Anda dengan AWS Serverless Application Model Pipeline (AWS SAM Pipeline). AWS SAM Pipeline adalah fitur baru AWS SAM CLI yang membantu Anda mengakses manfaat CI/CD dalam hitungan menit, seperti mempercepat frekuensi deployment, mempersingkat waktu tunggu untuk perubahan, dan mengurangi kesalahan deployment. AWS SAM Pipeline dilengkapi serangkaian templat alur default untuk AWS CodeBuild/CodePipeline sesuai praktik terbaik deployment AWS. Untuk informasi lebih lanjut dan untuk melihat tutorial, lihat blog Memperkenalkan AWS SAM Pipeline.

Alur untuk beberapa tim, cabang, dan Wilayah AWS

Untuk proyek besar, tidak jarang beberapa tim proyek mengerjakan komponen yang berbeda. Jika beberapa tim menggunakan repositori kode tunggal, maka dapat dipetakan sehingga setiap tim memiliki cabangnya sendiri. Cabang integrasi atau rilis untuk penggabungan akhir proyek juga harus ada. Jika menggunakan arsitektur layanan mikro atau berorientasi pada layanan, setiap tim bisa memiliki repositori kode sendiri.

Dalam skenario pertama, jika menggunakan alur tunggal, satu tim dapat memengaruhi kemajuan tim lain dengan memblokir alur. AWS menyarankan agar Anda membuat alur khusus untuk cabang tim dan alur rilis lainnya untuk pengiriman produk akhir.

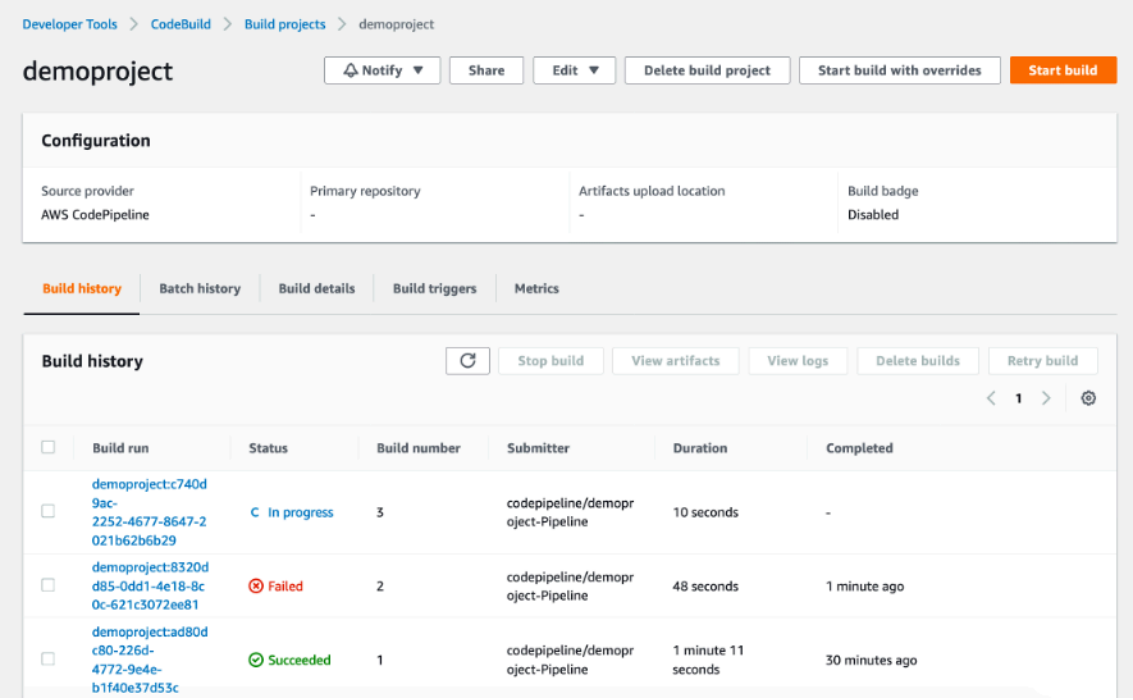
Integrasi alur dengan AWS CodeBuild

AWS CodeBuild dirancang untuk membantu organisasi Anda membangun proses pembangunan dengan ketersediaan tinggi dan skala hampir tak terbatas. AWS CodeBuild menyediakan lingkungan pemula untuk beberapa bahasa populer dan kemampuan untuk menjalankan kontainer Docker yang Anda tentukan.

Dengan manfaat integrasi ketat bersama AWS CodeCommit, AWS CodePipeline, dan AWS CodeDeploy, serta tindakan Lambda CodePipeline dan Git, alat CodeBuild sangat fleksibel.

Perangkat lunak dapat dibangun dengan menyertakan file `buildspec.yml` yang mengidentifikasi setiap langkah pembangunan, termasuk tindakan sebelum dan sesudah pembangunan, atau tindakan tertentu melalui alat CodeBuild.

Anda dapat melihat detail riwayat masing-masing pembangunan menggunakan dasbor CodeBuild. Peristiwa disimpan sebagai berkas log Amazon CloudWatch Logs.



Berkas log CloudWatch Logs di AWS CodeBuild

Integrasi alur dengan Jenkins

Anda dapat menggunakan alat pembangunan Jenkins untuk membuat alur pengiriman. Alur ini menggunakan tugas standar yang menentukan langkah-langkah untuk mengimplementasikan tahapan pengiriman berkelanjutan. Namun, pendekatan ini mungkin tidak optimal untuk proyek yang lebih besar karena status alur saat ini tidak sama antara mulai ulang Jenkins, implementasi persetujuan manual tidak mudah, dan pelacakan status alur yang kompleks dapat menjadi rumit.

Sebagai gantinya, AWS sebaiknya implementasikan pengiriman berkelanjutan dengan Jenkins menggunakan Plugin AWS Code Pipeline. Plugin ini memungkinkan penjabaran alur kerja kompleks menggunakan bahasa khusus domain seperti Groovy, dan dapat digunakan untuk mengatur alur yang kompleks. Fungsionalitas plugin AWS Code Pipeline dapat ditingkatkan menggunakan plugin satelit, seperti Plugin Tampilan Tahapan Alur, yang memvisualisasikan kemajuan tahapan terbaru

yang ditentukan dalam alur, atau Plugin Multicabang Alur, yang mengelompokkan build dari cabang yang berbeda.

AWS menyarankan untuk menyimpan konfigurasi alur dalam Jenkinsfile dan memeriksanya dalam repositori kode sumber. Hal ini memungkinkan pelacakan perubahan pada kode alur dan menjadi poin penting ketika bekerja dengan Plugin Multicabang Alur. AWS menyarankan untuk membagi alur menjadi beberapa tahap. Hal ini secara logis mengelompokkan langkah-langkah alur dan juga memungkinkan Plugin Tampilan Tahapan Alur untuk memvisualisasikan status terbaru alur.

Gambar berikut menunjukkan sampel alur Jenkins, dengan empat tahap yang sudah ditentukan dan divisualisasikan oleh Plugin Tampilan Tahapan Alur.

Tahapan alur Jenkins yang sudah ditentukan dan divisualisasikan oleh Plugin Tampilan Tahapan Alur

Metode deployment

Anda dapat menggunakan beberapa strategi dan variasi deployment untuk meluncurkan versi baru perangkat lunak dalam proses pengiriman berkelanjutan. Bagian ini membahas metode deployment yang paling umum: semua sekaligus (deploy di tempat), bergulir, tetap, dan biru/hijau. AWS menunjukkan metode mana yang didukung oleh AWS CodeDeploy dan AWS Elastic Beanstalk.

Tabel berikut merangkum karakteristik masing-masing metode deployment.

Metode	Dampak kegagalan deployment	Waktu deployment	Waktu henti nol	Tidak ada perubahan DNS	Pemrosesan rollback	Tujuan deploy kode
Deploy di tempat	Waktu henti	🕒	X	✓	Deploy ulang	Instans yang ada
Bergulir	Batch tunggal tidak bisa digunakan. Batch yang berhasil sebelum gagal menjalankan versi aplikasi baru.	🕒 🕒 †	✓	✓	Deploy ulang	Instans yang ada
Bergulir dengan batch tambahan (beanstalk)	Minimal jika batch pertama gagal, jika tidak, akan mirip	🕒 🕒 🕒 †	✓	✓	Deploy ulang	Instans baru dan yang sudah ada

Metode	Dampak kegagalan deployment	Waktu deployment	Waktu henti nol	Tidak ada perubahan DNS	Pemrosesan rollback	Tujuan deploy kode
	dengan bergulir.					
Tetap	Minimal	🕒 🕒 🕒 🕒	✓	✓	Deploy ulang	Instans baru
Lalu lintas pemisahan	Minimal	🕒 🕒 🕒 🕒	✓	✓	Rutekan ulang lalu lintas dan hentikan instans baru	Instans baru
Biru/hijau	Minimal	🕒 🕒 🕒 🕒	✓	×	kembali ke lingkungan lama	Instans baru

Semua sekaligus (deployment di tempat)

Semua sekaligus (deployment di tempat) adalah metode yang dapat Anda gunakan untuk meluncurkan kode aplikasi baru ke armada server yang ada. Metode ini menggantikan semua kode dalam satu tindakan deployment. Metode ini memerlukan waktu henti karena semua server dalam armada diperbarui sekaligus. Tidak perlu memperbarui catatan DNS yang ada. Jika deployment gagal, satu-satunya cara untuk memulihkan operasi adalah dengan men-deploy ulang kode pada semua server.

Dalam AWS Elastic Beanstalk deployment ini disebut Semua sekaligus, serta tersedia untuk aplikasi tunggal dan dengan beban seimbang. Dalam AWS CodeDeploy metode deployment ini disebut Deployment di tempat dengan konfigurasi deployment `AllAtOnce`.

Deployment bergulir

Dengan deployment bergulir, armada dibagi menjadi beberapa bagian sehingga semua armada tidak ditingkatkan dalam waktu bersamaan. Selama proses deployment, dua versi perangkat lunak, baru dan lama, berjalan pada armada yang sama. Metode ini memungkinkan pembaruan waktu henti nol. Jika deployment gagal, hanya bagian armada yang sudah diperbarui yang akan terpengaruh.

Variasi metode deployment bergulir, disebut peluncuran canary, pada awalnya melibatkan deployment versi perangkat lunak baru pada server dengan persentase yang sangat kecil. Dengan demikian, Anda dapat mengamati bagaimana perangkat lunak berperilaku dalam produksi pada beberapa server, sekaligus meminimalkan dampak perubahan yang dapat menyebabkan kegagalan. Jika ada peningkatan tingkat kesalahan dari deployment canary, perangkat lunak akan di-rollback. Jika tidak, persentase server dengan versi baru akan meningkat secara bertahap.

AWS Elastic Beanstalk telah mengikuti pola deployment bergulir dengan dua pilihan deployment, bergulir dan bergulir dengan batch tambahan. Pilihan-pilihan ini memungkinkan aplikasi untuk menaikkan skala terlebih dahulu sebelum menghentikan layanan server sehingga dapat mempertahankan kemampuan penuh selama deployment. AWS CodeDeploy menyelesaikan pola ini sebagai variasi deployment di tempat dengan pola seperti OneAtATime dan HalfAtATime.

Deployment tetap dan biru/hijau

Pola tetap menentukan deployment kode aplikasi dengan memulai rangkaian server baru dengan konfigurasi atau versi kode aplikasi baru. Pola ini memanfaatkan kemampuan cloud untuk membuat sumber daya server baru dengan panggilan API sederhana.

Strategi deployment biru/hijau adalah jenis deployment tetap yang juga mengharuskan pembuatan lingkungan lain. Setelah lingkungan baru siap dan lulus dari semua pengujian, lalu lintas akan dialihkan ke deployment baru ini. Yang terpenting, lingkungan lama, yaitu lingkungan “biru”, tetap siaga jika diperlukan rollback.

AWS Elastic Beanstalk mendukung pola deployment tetap dan biru/hijau. AWS CodeDeploy juga mendukung pola biru/hijau. Untuk informasi lebih lanjut tentang bagaimana layanan AWS mencapai pola tetap ini, baca laporan resmi Deployment Biru/Hijau di AWS .

Perubahan skema basis data

Perangkat lunak modern umumnya memiliki lapisan basis data. Yang biasa digunakan adalah basis data relasional, yang menyimpan data dan struktur data. Dalam proses pengiriman berkelanjutan, seringnya basis data perlu diubah. Menangani perubahan pada basis data relasional memerlukan pertimbangan khusus, dan sering menimbulkan tantangan yang berbeda yang muncul di proses deployment biner aplikasi. Saat meningkatkan biner aplikasi, Anda biasanya menghentikan aplikasi, meningkatkan, kemudian memulainya lagi. Anda tidak begitu memperhatikan status aplikasi, yang ditangani di luar aplikasi.

Saat meningkatkan basis data, Anda perlu mempertimbangkan status karena sebuah basis data dapat berisi banyak status dengan sedikit logika dan struktur.

Skema basis data sebelum dan sesudah penerapan perubahan harus dianggap sebagai versi basis data yang berbeda. Anda dapat menggunakan alat seperti Liquibase dan Flyway untuk mengelola versi.

Secara umum, alat-alat tersebut menggunakan beberapa varian metode berikut:

- Menambahkan tabel ke basis data yang digunakan untuk menyimpan versi basis data.
- Melacak perintah perubahan basis data dan mengumpulkannya dalam versi set perubahan. Pada Liquibase, perubahan ini disimpan dalam file XML. Flyway menggunakan metode yang sedikit berbeda, yaitu menangani set perubahan sebagai file SQL terpisah atau kadang-kadang sebagai kelas Java terpisah untuk transisi yang lebih kompleks.
- Saat diminta untuk meningkatkan basis data, Liquibase akan melihat tabel metadata dan menentukan set perubahan mana yang akan dijalankan untuk memperbarui basis data ke versi terbaru.

Ringkasan praktik terbaik

Berikut ini adalah beberapa praktik terbaik berupa anjuran dan larangan untuk CI/CD.

Saran:

- Kelola infrastruktur sebagai code (IaC)
  - Gunakan kontrol versi untuk kode infrastruktur Anda.
  - Manfaatkan sistem pelacakan/pelaporan bug.
  - Tinjau perubahan dengan rekan Anda sebelum menerapkannya.
  - Bangun desain/pola kode infrastruktur.
  - Uji perubahan infrastruktur seperti perubahan kode.
- Gabungkan developer ke dalam tim terpadu beranggotakan tidak lebih dari 12 orang.
- Minta semua developer mengirim pembaruan kode ke cabang utama sesering mungkin, tanpa cabang fitur yang lama berjalan.
- Adopsi sistem pembangunan seperti Maven atau Gradle di seluruh organisasi Anda dan membuat standar pembangunan secara konsisten.
- Minta developer membangun pengujian unit yang mencakup seluruh dasar kode.
- Pastikan durasi, jumlah, dan ruang lingkup pengujian unit mencakup 70% dari keseluruhan pengujian.
- Pastikan pengujian unit sudah diperbarui dan tidak terbengkalai. Kegagalan dalam pengujian unit harus diperbaiki, tidak diabaikan.
- Perlakukan konfigurasi pengiriman berkelanjutan sebagai kode.
- Tetapkan kontrol keamanan berbasis peran (yaitu, siapa yang dapat melakukan apa dan kapan).
  - Pantau/lacak setiap sumber daya yang ada.
  - Perhatikan layanan, ketersediaan, dan waktu respons.
  - Perhatikan, pelajari, dan tingkatkan.
  - Bagikan akses ke semua orang di tim.
  - Rencanakan metrik dan pemantauan ke dalam siklus hidup.
- Pertahankan dan lacak metrik standar.
  - Jumlah pembangunan.
  - Jumlah deployment.

    - Rata-rata waktu yang diperlukan perubahan untuk mencapai produksi.
    - Rata-rata waktu dari tahap alur pertama ke masing-masing tahap.
    - Jumlah perubahan yang mencapai produksi.
    - Rata-rata waktu pembangunan.
- Gunakan beberapa alur yang berbeda untuk setiap cabang dan tim.

Hindari:

- Adanya cabang yang sudah berjalan lama dengan penggabungan besar yang rumit.
- Pengujian manual.
- Proses persetujuan, gate, tinjauan kode, dan tinjauan keamanan manual.

Kesimpulan

Integrasi berkelanjutan dan pengiriman berkelanjutan memberikan skenario yang ideal untuk tim aplikasi organisasi Anda. Developer hanya perlu mendorong kode ke repositori. Kode ini akan diintegrasikan, diuji, di-deploy, diuji kembali, digabungkan dengan infrastruktur, melalui ulasan keamanan dan kualitas, dan siap untuk men-deploy dengan kualitas terbaik.

Dengan CI/CD, kualitas kode meningkat dan pembaruan perangkat lunak dapat dikirim dengan cepat dengan jaminan tidak akan ada perubahan yang menyebabkan kegagalan. Dampak dari setiap rilis dapat berkorelasi dengan data dari produksi dan operasi. CI/CD juga dapat digunakan untuk merencanakan siklus berikutnya—praktik DevOps penting dalam transformasi cloud di organisasi Anda.

Sumber bacaan lebih lanjut

Untuk informasi lebih lanjut tentang topik yang dibahas dalam laporan resmi ini, lihat laporan resmi AWS berikut:

- Gambaran Umum Opsi Deployment di AWS
- Deployment Biru/Hijau di AWS
- Menyiapkan alur CI/CD dengan mengintegrasikan Jenkins dengan AWS CodeBuild dan AWS CodeDeploy
- Layanan mikro di AWS
- Docker di AWS: Menjalankan Kontainer di Cloud

Kontributor

Individu dan organisasi berikut berkontribusi pada dokumen ini:

- Amrish Thakkar, Kepala Arsitek Solusi (Principal Solutions Architect), AWS
- David Stacy, Konsultan Senior (Senior Consultant) - DevOps, Layanan Profesional AWS
- Asif Khan, Arsitek Solusi (Solutions Architect), AWS
- Xiang Shen, Arsitek Solusi Senior (Senior Solutions Architect), AWS

Revisi dokumen

Untuk mendapatkan notifikasi tentang pembaruan laporan resmi ini, silakan berlangganan umpan RSS.

perubahan-riwayat-pembaruan	deskripsi-riwayat-pembaruan	tanggal-riwayat-pembaruan
Publikasi awal	Laporan resmi pertama kali dipublikasikan	27 Oktober 2021
Publikasi awal	Laporan resmi pertama kali dipublikasikan	1 Juni 2017

Pemberitahuan

Pelanggan bertanggung jawab untuk membuat penilaian independen mereka sendiri atas informasi dalam dokumen ini. Dokumen ini: (a) hanya disediakan sebagai informasi, (b) berisi penawaran produk dan praktik AWS saat ini, yang dapat berubah tanpa pemberitahuan, dan (c) tidak menjadi komitmen atau jaminan apa pun dari AWS dan afiliasi, pemasok, atau pemberi lisensinya. Produk atau layanan AWS disediakan “sebagaimana adanya” tanpa jaminan, representasi, atau ketentuan apa pun, baik tersurat maupun tersirat. Tanggung jawab dan kewajiban AWS kepada pelanggannya dikendalikan oleh perjanjian AWS, dan dokumen ini bukan bagian dari, juga tidak mengubah, perjanjian apa pun antara AWS dan pelanggannya.